**IT 1090C Computer Programming I   
Prof. Tom Wulf**

Lab 13: File ListMaker  
**20 Points**

# Directions:

Project: Lab\_13\_FileListMaker

* You will refactor our Lab 11 Listmaker lab. (For your convenience, I have attached the lab 11 directions here at the end of this doc. You should not have to redo Lab 11…)
* Instead of a single list held in memory, the new refactored program will be able to save and load lists from disk.

In addition to these existing options:  
A – Add an item to the list  
D – Delete an item from the list

P - Print the list to the screen Change this to V for view

Q – Quit the program  
  
You will add:  
O – Open a list file from disk  
S – Save the current list file to disk  
C – Clear removes all the elements from the current list  
V - CHANGE THE P Print OPTION to V for View

* You might notice that there is no explicit option for creating a new list. To do this, we merely use the add option to build the list in memory.
* The big challenge here is to keep track of the program state:
  + Use a boolean variable like needsToBeSaved to keep track of list edits. (Traditionally this has been called a ‘dirty’ flag. The file becomes ‘dirty’ when it needs to be saved.
  + If the user loads a list it does not need to be saved until it is changed by adding or deleting items. (The user could load a list only to view it and then would want to load another in its place without saving…)
  + If the user begins to build a list by adding items and does not load an existing list keep track of this new list. Prompt the user on exit to save the list or abandon it.
  + Similarly, prompt the user to save an unsaved list before loading a new list from disk.. etc.
  + Loaded lists are always saved with the same filename.
  + All list files have the .txt extension

1. Name your project: Lab13\_FileListMaker
2. Implement and thoroughly test your refactored code.
3. Provide a series of screen shots that document and establish that your program:  
     
   - Will load a list from disk

– Will save a list loaded from disk that has been modified by deleting or adding items using the same filename so it overwrites the file. Reload the changed file to clearly show that you were able to save the changes

- Will allow the user to create a new list and save it to disk with a provided basename. List files should have the .txt extension.

- bullet proof the program so that a user can’t loose data:

- if they ask to load a file but have an unsaved list in memory, prompt them to first save the list.

- if they go to quit the program and have an unsaved list prompt them

Text

Description automatically generated  
  
Submission:

Submit this file **Lastname\_Firstname\_Lab\_13\_FileListMaker.docx** using your name. Then submit either your GitHub repo link or your complete project archive of your IntelliJcolde. **Lastname\_Firstname\_Lab\_13\_FileListMaker.zip.**

THIS IS THE PREVIOUS LAB THAT YOU ARE REFACTORING AND IS HERE FOR YOUR CONVENIENCE ONLY. DON’T DO THIS!!!  
 **IT 1090C Computer Programming I   
IT 6090C Java Programming**

**Prof. Tom Wulf  
Lab 11: ListMaker**

**20 pts**  
  
Learning Goals:

* Work with the ArrayList to create a dynamic list
* The user can Add and item, Remove an item, Print the list of items, or Quit
* Use the SafeInput library we created for the input

Directions:

1. Start by creating a menu driven loop:
   1. The Loop awaits user input repeating until the Quit command is issued
   2. Options: (Details of the options are below)  
      A – Add an item to the list  
      D – Delete an item from the list

P - Print the list to the screen

Q – Quit the program

* 1. The program gets one of these commands from the user and executes that function
  2. Initially stub out the functions so you have a program that you can run almost immediately as you develop it. You will develop each of the menu options as a separate java method.
  3. You also will need some utility functions. These should be private static methods located in the same file as your java main().
     1. You will want to display the current list along with the menu of options so the user can see what they are doing. (Just like we display the current board in Tic Tac Toe each turn…)
     2. You also need to display the menu or options and prompt. Some analysis shows that we will need to display the list each time prior to displaying the menu.
     3. You need to display a numbered version of the list to allow users to pick list elements for deletion. Here the user looks at the display and then indicates the item to delete by the number. You should use 1 based indexing for the user (items begin at 1) and convert it to java indexing by subtracting one since indexing is zero based. (Again, this is like the conversion of the board coordinates in Tic Tac Toe from 1-3 to 0-2.) You can also use this method to display the list to the user for the menu Print option.
  4. Option Details:  
     **Add** - prompt the user for a String entry for the list. Make sure the String is not empty (i.e. zero-length) add the new list item to the end of the list.  
       
     **Delete** – if there are no items in the list, display a status msg that the list is empty display the list items with a numbered 1 – based index (numbers start at 1)  
     Make sure that the user provides a valid index. Convert the index to a valid array zero based one and remove the item from the list. Give the user an escape option so they can decide not to delete a list item. A simple way to do this is to simply return without deleting anything if the user enters an invalid index. (So, you show the user the current list and they decide not to delete anything, they enter an invalid index to do so.  
       
     **Print -** – if there are no items in the list, display a status msg that the list is empty display the list items with a numbered 1 – based index (numbers start at 1) Note that you can use the same list display for the Print option as for the Delete option.  
       
     **Quit** – before exiting, warn the user that the current list will not be saved and is gone once they exit. Let them confirm that they want to actually quit.

1. Test your code thoroughly. Provide output screen shots in pairs for each of the menu functions that clearly show that you completed them and that they made the correct effect on the list. Try to show this through a readonable sequence. Show several items being added to the list and then remove several.